Ace Lorence L. Lapan February 22, 2024

BSIT-32E1 Mr. Alamo

Object-oriented programming (OOP) is vital in software development due to its ability to enhance code organization, reusability, and maintainability. The core principles of OOP encapsulation, inheritance, polymorphism, and abstraction provide a structured approach to design. Encapsulation bundles data and methods, shielding internal complexities. Inheritance fosters code reuse, while polymorphism allows treating diverse objects uniformly. Abstraction hides implementation details, simplifying interactions. OOP fosters modular design, easing collaboration and supporting flexible, extensible systems. Its benefits, such as code reusability and maintainability, make it a cornerstone in developing scalable, well-structured software applications.

**Encapsulation:**

Example: Consider a class representing a bank account. Encapsulation allows you to encapsulate attributes like balance and methods like deposit and withdraw within the class. This shields the internal details of the implementation from the external world.

**Inheritance:**

Example: In a scenario where you have different types of vehicles, you can have a superclass called Vehicle with common attributes and methods. Subclasses like Car and Motorcycle can inherit from Vehicle and add their specific features.

**Polymorphism:**

Example: In a graphical application, a Shape class might have a method draw. Various subclasses like Circle and Rectangle can implement their version of the draw method. Polymorphism allows treating them uniformly, like calling draw on any shape object.

**Abstraction:**

Example: Consider a remote control. The user interacts with the remote (abstract interface) without needing to know the internal details of how it sends signals to the TV. The remote serves as an abstraction of the TV control process.